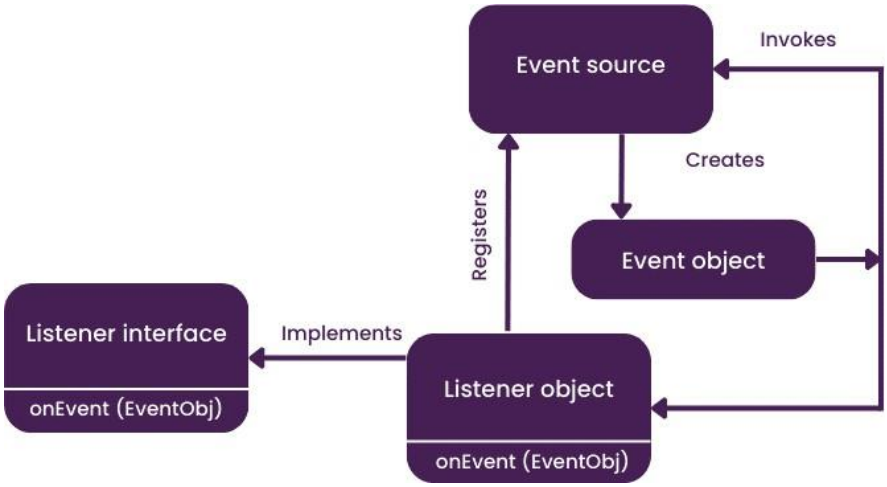
## UNIT V – MULTITHREADING, GUI PROGRAMMING & EVENT HANDLING

### Event Delegation Model:

- The Event Delegation Model (EDM) is a mechanism in Java (introduced with Java 1.1) to handle events in GUI programming (AWT & Swing).
- Instead of having every component handle its own events, the event delegation model separates the event source from the event listener.
- This makes event handling cleaner, flexible, and more object-oriented.




### Components of the Event Delegation Model

1. **Event Source**
   - The GUI component that generates events when a user interacts with it.
   - Example: Button, TextField, Checkbox.
   - The event source is responsible for:
     - ✓ Creating an Event Object when an action happens.
     - ✓ Invoking the registered listener's callback method.
2. **Event Object**
   - When an action occurs, the Event Source creates an Event Object.
   - The Event Object carries information about the event (e.g., mouse click position, button label, key pressed.
   - This Event Object is then passed to the Listener Object.
   - Example classes:
     - ✓ ActionEvent → Button click
     - ✓ MouseEvent → Mouse operations
     - ✓ KeyEvent → Keyboard operations
     - ✓ WindowEvent → Closing/minimizing a window

**3. Listener Interface**

- Defines the contract for handling a specific type of event.
- Example interfaces: ActionListener, MouseListener, KeyListener.
- Contains one or more abstract methods (like actionPerformed, mouseClicked) that must be implemented.
- Example interfaces:
  - ✓ ActionListener → actionPerformed(ActionEvent e)
  - ✓ MouseListener → mouseClicked(MouseEvent e)
  - ✓ KeyListener → keyPressed(KeyEvent e)

**4. Listener Object**

- A concrete class that implements the Listener Interface.
- It contains the actual callback methods that define what should happen when an event occurs.
- Example:

```
class MyHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Button Clicked!");
    }
}
```

**Working of Event Delegation Model**

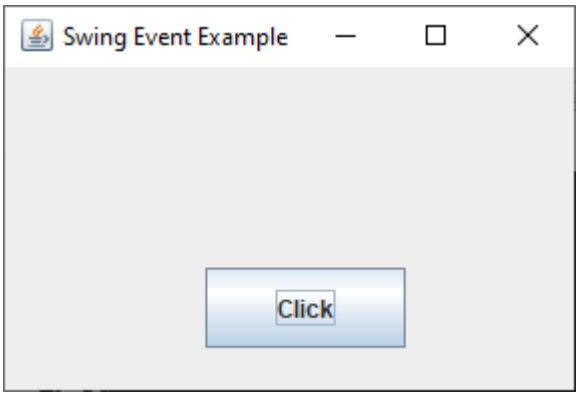
- User performs an action (click, press a key, move mouse).
- Event Source generates an Event Object.
- Event Source notifies all registered Event Listeners.
- The listener's callback method executes to handle the event.

**Example Program:**

```
import javax.swing.*;
import java.awt.event.*;
public class SwingEDM
{
   public static void main(String[] args) {
      JFrame f = new JFrame("Swing Event Example");
      JButton b = new JButton("Click");
      b.setBounds(100, 100, 100, 40);
      // Registering an ActionListener
      b.addActionListener(e ->JOptionPane.showMessageDialog(f, "Button Clicked!"));
```

```
        f.add(b);
        f.setSize(300, 200);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

**Output:**





**ActionListener:**

- ActionListener is an interface in the java.awt.event package.
- It is used to handle action events such as:
  - ✓ Clicking a button (JButton)
  - ✓ Choosing a menu item (JMenuItem)
  - ✓ Pressing Enter in a text field (JTextField)
- In short, whenever the user performs an action event, the ActionListener's actionPerformed() method is executed.

**Syntax:**

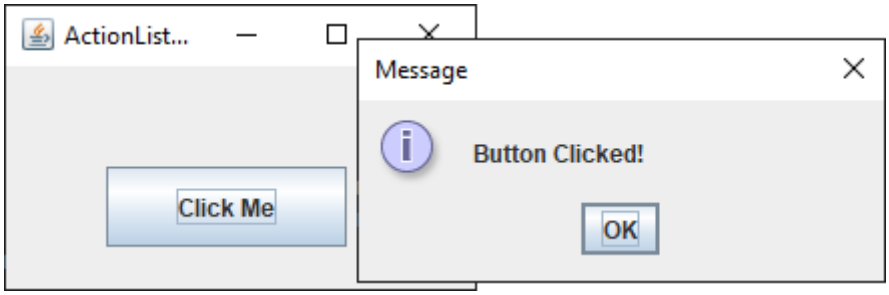
```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public interface ActionListener extends EventListener
{
    void actionPerformed(ActionEvent e);
}
```

**Steps to use ActionListener:**

- Create a GUI component (e.g., JButton).
- Implement ActionListener interface or use an anonymous class/lambda.
- Override the actionPerformed(ActionEvent e) method.
- Register the listener using addActionListener() method.

**Example:**

```
import javax.swing.*;
import java.awt.event.*;
public class ActionListenerExample implements ActionListener
{
    JButton button;
    ActionListenerExample()
    {
        JFrame frame = new JFrame("ActionListener Example");
        button = new JButton("Click Me");
        button.setBounds(50, 50, 120, 40);
        button.addActionListener(this);     // Register ActionListener
        frame.add(button);
        frame.setSize(250, 150);
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    // This method is called when the button is clicked
    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(null, "Button Clicked!");
    }
    public static void main(String[] args)
    {
        new ActionListenerExample();
    }
}
```




**Mouse Listener:**

- MouseListener is an interface in java.awt.event package.
- It is used to handle mouse events like:
  - ✓ Mouse clicked
  - ✓ Mouse pressed
  - ✓ Mouse released

- ✓ Mouse entered (cursor enters a component area)
- ✓ Mouse exited (cursor leaves a component area)

**Syntax:**

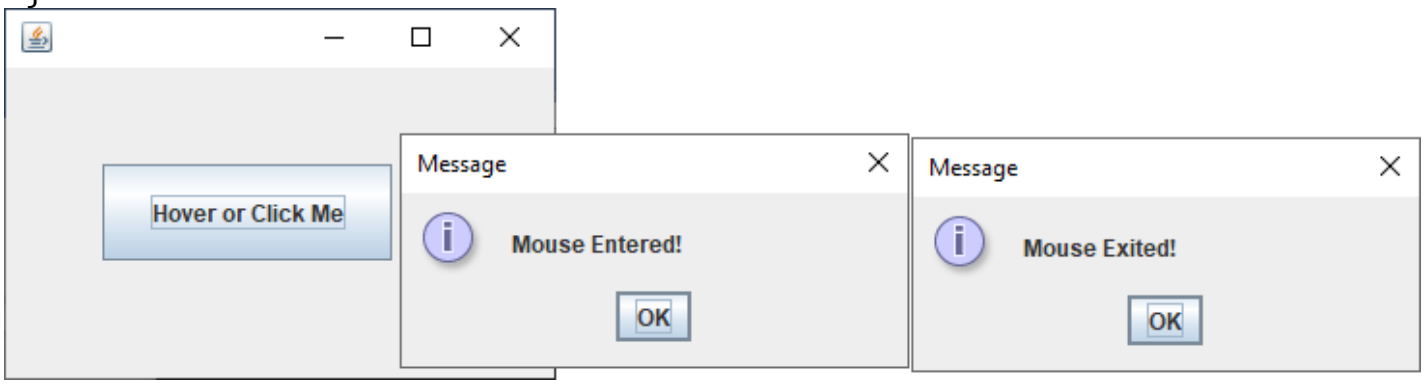
```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;
public interface MouseListener extends EventListener
{
    void mouseClicked(MouseEvent e);
    void mousePressed(MouseEvent e);
    void mouseReleased(MouseEvent e);
    void mouseEntered(MouseEvent e);
    void mouseExited(MouseEvent e);
}
```

**Steps to Use MouseListener:**

- ✓ Create a component (e.g., JButton, JPanel, or JLabel).
- ✓ Implement MouseListener interface.
- ✓ Override all five methods.
- ✓ Register the listener using addMouseListener().
- ✓ (400, 300); setLayoutimport javax.swing.*;

```
import java.awt.event.*;
public class MouseListenerDialogExample extends JFrame implements MouseListener
{
   JButton button;
   MouseListenerDialogExample()
   {
      button = new JButton("Hover or Click Me");
      button.setBounds(100, 100, 200, 50);
      // Register MouseListener
      button.addMouseListener(this);
      add(button);
      setSize (null);
      setVisible(true);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
   }
   // Implement all methods with dialog boxes
   public void mouseClicked(MouseEvent e)
   {
      JOptionPane.showMessageDialog(this, "Mouse Clicked!");
   }
   public void mousePressed(MouseEvent e)
   {
      JOptionPane.showMessageDialog(this, "Mouse Pressed!");
```

```
    }
    public void mouseReleased(MouseEvent e)
    {
        JOptionPane.showMessageDialog(this, "Mouse Released!");
    }
    public void mouseEntered(MouseEvent e)
    {
        JOptionPane.showMessageDialog(this, "Mouse Entered!");
    }
    public void mouseExited(MouseEvent e)
    {
        JOptionPane.showMessageDialog(this, "Mouse Exited!");
    }
    public static void main(String[] args)
    {
        new MouseListenerDialogExample();
    }
}
```




**Building basic forms and interfaces with buttons, labels, text fields:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class SimpleForm
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Simple Form");   // Create a frame
        frame.setSize(300, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout()); // Using FlowLayout
        JLabel label = new JLabel("Enter your name:");  // Create components
        JTextField textField = new JTextField(15);
        JButton button = new JButton("Submit");
        button.addActionListener(new ActionListener()  // Add action listener for button
        {
```

```
            public void actionPerformed(ActionEvent e)
            {
                String name = textField.getText();
                JOptionPane.showMessageDialog(frame, "Hello, " + name + "!");
            }
        });

        frame.add(label);  // Add components to frame
        frame.add(textField);
        frame.add(button);
        frame.setVisible(true); // Make frame visible
    }
}
```